ChainLight

Rhinestone Safe7579 Security Audit

: Safe7579 - An ERC-7579 adapter for Safe accounts

Apr 15, 2025

Revision 1.1

ChainLight@Theori

Table of Contents

Executive Summary

Beginning on March 19, 2025, ChainLight of Theori conducted a three-day security audit of Rhinestone's Safe7579 contracts. The primary goal of the audit was to identify critical security vulnerabilities and evaluate potential impacts.

Summary of Findings

The audit revealed a total of four issues, categorized by severity as follows:

- **Low:** 2 issues
- **Informational:** 2 issues

We also separately reported two out-of-scope issues in related smart contracts, which were not included in this report.

Audit Overview

Scope

Name	Rhinestone Safe7579 Security Audit
Target / Version	• Git Repository (rhinestonewtf/safe7579): commit `5cac8be077b8e207969322e115416fa203092b49` (changes since previous audit), the commit id after the patches is `f314e65b965794160e36347457271688da112e0a`
Application Type	Smart contracts
Lang. / Platforms	Smart contracts [Solidity]

Code Revision

N/A

Severity Categories

Severity	Description
Critical	The attack cost is low (not requiring much time or effort to succeed in the actual attack), and the vulnerability causes a high-impact issue. (e.g., Effect on service availability, Attacker taking financial gain)
High	An attacker can succeed in an attack which clearly causes problems in the service's operation. Even when the attack cost is high, the severity of the issue is considered "high" if the impact of the attack is remarkably high.
Medium	An attacker may perform an unintended action in the service, and the action may impact service operation. However, there are some restrictions for the actual attack to succeed.
Low	An attacker can perform an unintended action in the service, but the action does not cause significant impact or the success rate of the attack is remarkably low.
Informational	Any informational findings that do not directly impact the user or the protocol.
Note	Neutral information about the target that is not directly related to the project's safety and security.

Status Categories

Status	Description
Reported	ChainLight reported the issue to the client.
WIP	The client is working on the patch.
Patched	The client fully resolved the issue by patching the root cause.
Mitigated	The client resolved the issue by reducing the risk to an acceptable level by introducing mitigations.
Acknowledged	The client acknowledged the potential risk, but they will resolve it later.
Won't Fix	The client acknowledged the potential risk, but they decided to accept the risk.

Finding Breakdown by Severity

Category	Count	Findings
Critical	0	• N/A
High	0	• N/A
Medium	0	• N/A
Low	2	• `SAFE7579-001` • `SAFE7579-003`
Informational	2	• `SAFE7579-002` • `SAFE7579-004`
Note	0	• N/A

Findings

Summary

#	ID	Title	Severity	Status
1	`SAFE7579-001`	Edge Case of Address Check in `ModuleManager._isHookInstalled()`	Low	Patched
2	`SAFE7579-002`	Unused Argument in `withHook()` and `tryWithHook()`	Informational	Patched
3	`SAFE7579-003`	`Safe7579.installModule()` Should Verify Module Type	Low	Patched
4	`SAFE7579-004`	Minor Suggestions	Informational	Patched

#1 `SAFE7579-001` Edge Case of Address Check in `ModuleManager._isHookInstalled()`

ID	Summary	Severity
`SAFE7579-001`	The `ModuleManager._isHookInstalled()` function incorrectly returns `true` when both the hook and `module` addresses are set to `address(0)`. This behavior mistakenly recognizes the zero address (`address(0)`) as an installed hook.	Low

Description

In the `ModuleManager` contract, the `_isHookInstalled()` function verifies whether the hook address obtained from `globalHook[msg.sender]` (via the `getActiveHook()` function) matches the provided `module` address. Consequently, when both the hook and `module` addresses are set to `address(0)`, the function erroneously identifies the zero address as an installed hook.

Impact

Low

The issue's impact is limited, as it cannot be abused to falsely indicate an arbitrary address as a installed module.

Recommendation

Modify the `_isHookInstalled()` function to explicitly return `false` when the hook is set to `address(0)`. This ensures the zero address cannot be mistaken as an installed hook.

Remediation

Patched

The issue has been resolved by updating `_isHookInstalled()` to include the condition `module != address(0) &&`, effectively preventing `address(0)` from being recognized as an installed hook.

#2 `SAFE7579-002` Unused Argument in `withHook()` and `tryWithHook()`

ID	Summary	Severity
`SAFE7579-002`	Due to changes in pre-validation hooks, the `selector` argument in `withHook()` and `tryWithHook()` became obsolete and should be removed.	Informational

Description

Previously, the `withHook()` and `tryWithHook()` modifiers utilized the `selector` argument to execute a function signature specific hook (`sigHook`). This logic was removed, making the `selector` argument redundant. Additionally, comments within the codebase continue to imply the existence of this now non-existent hook, creating potential confusion.

Impact

Informational

Although there is no immediate impact on functionality, retaining an unused argument along with outdated comments may cause confusion for developers tasked with maintaining or auditing the codebase.

Recommendation

Remove the unused `selector` argument from both the `withHook()` and `tryWithHook()` methods, and update the corresponding code comments to accurately reflect current functionality.

Remediation

Patched

It has been patched as recommended.

#3 `SAFE7579-003` `Safe7579.installModule()` Should Verify Module Type

ID	Summary	Severity
`SAFE7579-003`	`Safe7579.installModule()` does not verify that the provided module matches the expected module type.	Low

Description

The function `Safe7579.installModule()` installs a given module as a specified type without performing any validation of the module's actual type. This may result in functionality issues or other unintended behaviors if a module is installed under an incorrect type.

Impact

Low

While this issue can lead to functionality issues or other unintended behaviors (with low likelihood), triggering this issue requires privileged access.

Recommendation

Add a check within `Safe7579.installModule()` verifying the return value of `IModule(...).isModuleType(...)` to ensure the installed module matches the expected type.

Remediation

Patched

A `withCorrectModuleType` modifier performing the recommended check has been implemented.

#4 `SAFE7579-004` Minor Suggestions

ID	Summary	Severity
`SAFE7579-004`	The description includes multiple suggestions for preventing incorrect settings caused by operational mistakes, mitigating potential issues, and improving code maturity and readability.	Informational

Description

Code Maturity

1. The comment in `ModuleManager._callFallbackHandler()` is ambiguous regarding what "default handler" refers to. Since it only checks the function signature for ERC721/1155 without a separate default handler address, the comment should be updated accordingly.

Impact

Informational

Recommendation

Consider applying the suggestions in the description above.

Remediation

Patched

It has been patched as recommended.

Revision History

Version	Date	Description
1.0	Apr 11, 2025	Initial version
1.1	Apr 15, 2025	Commit ID information update

ChainLight